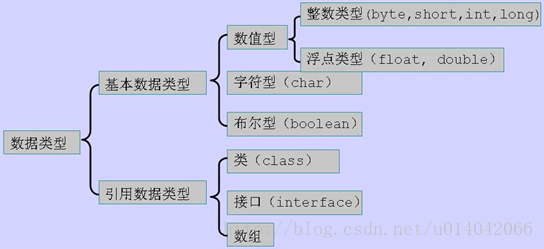
# Java面试题

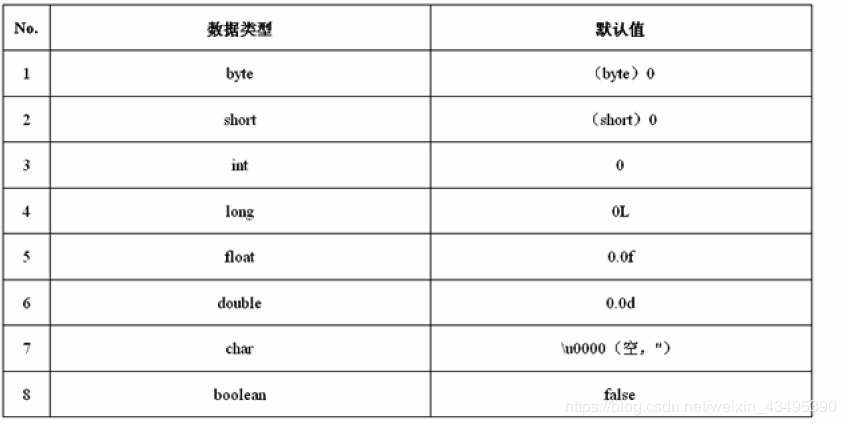
## Java基础

### Java中的数据类型

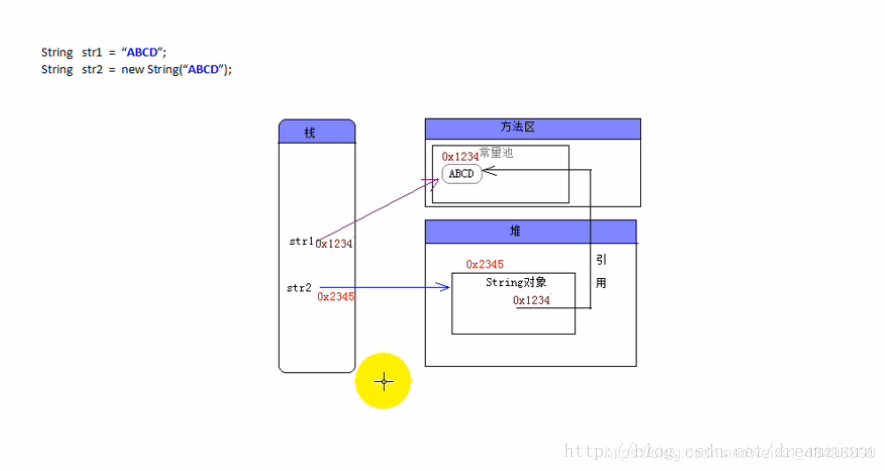


下图单位是bit,非字节 1B=8bit





### ****String能被继承吗？为什么？****

不可以，因为String类有final修饰符，而final修饰的类是不能被继承的，实现细节不允许改变。平常我们定义的String str=”abc”(直接赋一个字面量);其实和String str=new String(“abc”)(通过构造器构造)还是有差异的。  


### String str=“abc”和String str=new String(“abc”); 产生几个对象？

1.前者1或0，后者2或1，先看字符串常量池，如果字符串常量池中没有，都在常量池中创建一个，如果有，前者直接引用，后者在堆内存中还需创建一个“abc”实例对象。

2.对于基础类型的变量和常量：变量和引用存储在栈中，常量存储在常量池中。

3.为了提升jvm（JAVA虚拟机）性能和减少内存开销，避免字符的重复创建 项目中还是不要使用new String去创建字符串，最好使用String直接赋值。

### ****String， Stringbuffer， StringBuilder 的区别。****

#### String

字符串常量(final修饰，不可被继承)。String是常量，当创建之后即不能更改。(可以通过StringBuffer和StringBuilder创建String对象(常用的两个字符串操作类)。

#### **StringBuffer**

字符串变量（线程安全）,其也是final类别的，不允许被继承，其中的绝大多数方法都进行了同步处理，包括常用的Append方法也做了同步处理(synchronized修饰)。其自jdk1.0起就已经出现。其toString方法会进行对象缓存，以减少元素复制开销。

public synchronized String toString() {

if (toStringCache == null) {

toStringCache = Arrays.copyOfRange(value, 0, count);

}

return new String(toStringCache, true);}

#### StringBuilder

字符串变量（非线程安全）,其自jdk1.5起开始出现。与StringBuffer一样都继承和实现了同样的接口和类，方法除了没使用synch修饰以外基本一致，不同之处在于最后toString的时候，会直接返回一个新对象。

public String toString() {

// Create a copy, don’t share the array

return new String(value, 0, count);}

### 讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候， 他们的执行顺序。

此题考察的是类加载器实例化时进行的操作步骤（加载–>连接->初始化）。

父类静态变量、  
父类静态代码块、  
子类静态变量、  
子类静态代码块、  
父类非静态变量（父类实例成员变量）、  
父类构造函数、  
子类非静态变量（子类实例成员变量）、  
子类构造函数。

### 抽象类和接口的区别，类可以继承多个类么，接口可以继承多个接口么,类可以实现多个接口么。

1、抽象类和接口都不能直接实例化，如果要实例化，抽象类变量必须指向实现所有抽象方法的子类对象，接口变量必须指向实现所有接口方法的类对象。  
2、抽象类要被子类继承，接口要被类实现。  
3、接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现  
4、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。  
5、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个类实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。  
6、抽象方法只能申明，不能实现。abstract void abc();不能写成abstract void abc(){}。  
7、抽象类里可以没有抽象方法  
8、如果一个类里有抽象方法，那么这个类只能是抽象类  
9、抽象方法要被实现，所以不能是静态的，也不能是私有的。  
10、接口可继承接口，并可多继承接口，但类只能单根继承。

### **final 的用途** 类、变量、方法 final 修饰的类叫最终类，该类不能被继承。 final 修饰的方法不能被重写。 final 修饰的变量叫常量，常量必须初始化，初始化之后值就不能被修改。

### 访问修饰符 public、private、protected、default 的作用域



#### public

Java语言中访问限制最宽的修饰符，一般称之为“公共的”。被其修饰的类、属性以及方法不仅可以跨类访问，而且允许跨包（package）访问。

#### private

Java语言中对访问权限限制的最窄的修饰符，一般称之为“私有的”。被其修饰的类、属性以及方法只能被该类的对象访问，其子类不能访问，更不能允许跨包访问。

#### protect

介于public 和 private 之间的一种访问修饰符，一般称之为“保护形”。被其修饰的类、属性以及方法只能被类本身的方法及子类访问，即使子类在不同的包中也可以访问。

#### Default

即不加任何访问修饰符，通常称为"默认访问模式"。该模式下，只允许在同一个包中进行访问。

### &和&&的区别？

虽然二者都要求运算符左右两端的布尔值都是true整个表达式的值才是true。&&之所以称为短路运算是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是null而且不是空字符串，应当写为：username != null &&!username.equals(“”)，二者的顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的equals比较，否则会产生NullPointerException异常。注意：逻辑或运算符（|）和短路或运算符（||）的差别也是如此。

### 请列出5个运行时异常

NullPointerException     空指针异常

ClassCastException      类型转换异常

IllegalArgumentException     非法参数异常

IndexOutOfBoundsException   下标越界异常

ArrayIndexOutOfBoundsException 数组下标越界

### == 和 equals 的区别是什么？

#### == 解读

对于基本类型和引用类型 == 的作用效果是不同的，如下所示：  
 基本类型：比较的是值是否相同；  
 引用类型：比较的是引用是否相同；

#### equals

equals 默认情况下是引用比较，只是很多类重新了 equals 方法，比如 String、Integer 等把它变成了值比较，所以一般情况下 equals 比较的是值是否相等。

### String 类的常用方法都有那些？

indexOf()：返回指定字符的索引。

charAt()：返回指定索引处的字符。

replace()：字符串替换。

trim()：去除字符串两端空白。

split()：分割字符串，返回一个分割后的字符串数组。

getBytes()：返回字符串的 byte 类型数组。

length()：返回字符串长度。

toLowerCase()：将字符串转成小写字母。

toUpperCase()：将字符串转成大写字符。

substring()：截取字符串。

equals()：字符串比较。

### instanceof 关键字

instanceof 严格来说是Java中的一个双目运算符，用来测试一个对象是否为一个类的实例，用法为：

boolean result = obj instanceof Class

String类中的equals方法就有判断传入的对象是否为String对象。

